Robotics Nanodegree

Deep Learning Project

In this project, we'll use the deep neural network to identify and track a target in simulation. So-called "follow me" applications like this are key to many fields of robotics and the very same techniques you apply here could be extended to scenarios like advanced cruise control in autonomous vehicles or human-robot collaboration in industry.

If you need further information, assistance or referral about a project issue, please contact kiang.ng@hotmail.com.

Ng Fang Kiang
Kiang.ng@hotmail.com
Instagram.com/jorcus96
linkedin.com/in/jorcus
fb.com/ngfangkiang
github.com/jorcus

Ng Fang Kiang

Project 4: Follow Me

Prepared/Updated : 25 Oct 201

Deep Learning Project

In this project, we will train a deep neural network to identify and track a target in simulation. So-called "follow me" applications like this are key to many fields of robotics and the very same techniques you apply here could be extended to scenarios like advanced cruise control in autonomous vehicles or human-robot collaboration in industry.

Network Architecture

I initiate with the 2 encoder layers, 1x1 convolution and then 2 decoder layers and it fails to detect the target when its far away from the quadrotor. Then, I increase each encoder and decorder layers, and finalize the model architecture as 3 encoder layers, the 1x1 convolution, and then 3 decoder layers as following...

```
def fcn_model(inputs, num_classes):

    # TODO Add Encoder Blocks.
    # Remember that with each encoder layer, the depth of your model (the number of filters) increases.
    layer_1 = encoder_block(inputs, 32, 2)
    layer_2 = encoder_block(layer_1, 64, 2)
    layer_3 = encoder_block(layer_2, 128, 2)

    # TODO Add 1x1 Convolution layer using conv2d_batchnorm().
    layer_4 = conv2d_batchnorm(layer_3, 256, kernel_size=1, strides=1)

    # TODO: Add the same number of Decoder Blocks as the number of Encoder Blocks
    layer_5 = decoder_block(layer_4, layer_2, 128)
    layer_6 = decoder_block(layer_5, layer_1, 64)
    layer_7 = decoder_block(layer_6, inputs, 32)

    # The function returns the output layer of your model. "layer_5" is the final layer obtained from the
 last decoder_block()
    return layers.Conv2D(num_classes, 1, activation='softmax', padding='same')(layer_7)
```

Hyperparameters

I've tuned a few scenario of hyperparameters and this is my final decision because it is able to detect an object that is far away from the vision.

Learning rate : 0.002 (From my observation when tuning the learning rate, It resulted the neural networks learned faster when higher learning rate is implemented. Due to I'm going to use high numbers of learning rate, so I a minor change increased from 0.001 to 0.002.)

Batch size: 32 (Stay Default)

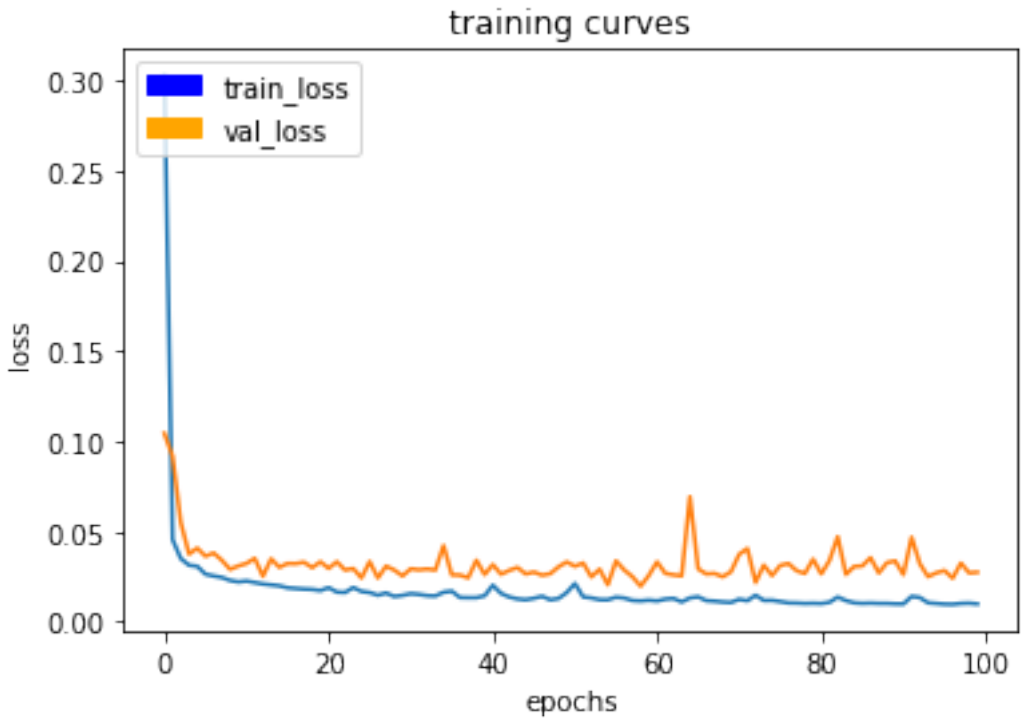
Numbers of Epoch : 100 (This numbers of approached is abit high, it might cause overfitting. However, I still decided to select this because I'm going to modified this project when I have time)

Steps per Epoch : 200 (Stay Default)

Validation Steps : 50 (Stay Default)

Workers : 5

Result




```
# Scores for while the quad is following behind the target.
true_pos1, false_pos1, false_neg1, iou1 = scoring_utils.score_run_iou(val_following, pred_following)
```

```
number of validation samples intersection over the union evaulated on 542
average intersection over union for background is 0.995990646239
average intersection over union for other people is 0.36794329476
average intersection over union for the hero is 0.916386351051
number true positives: 539, number false positives: 0, number false negatives: 0
```

```
# Scores for images while the quad is on patrol and the target is not visable
true_pos2, false_pos2, false_neg2, iou2 = scoring_utils.score_run_iou(val_no_targ, pred_no_targ)
```

```
number of validation samples intersection over the union evaulated on 270
average intersection over union for background is 0.986185149222
average intersection over union for other people is 0.718848343485
average intersection over union for the hero is 0.0
number true positives: 0, number false positives: 63, number false negatives: 0
```

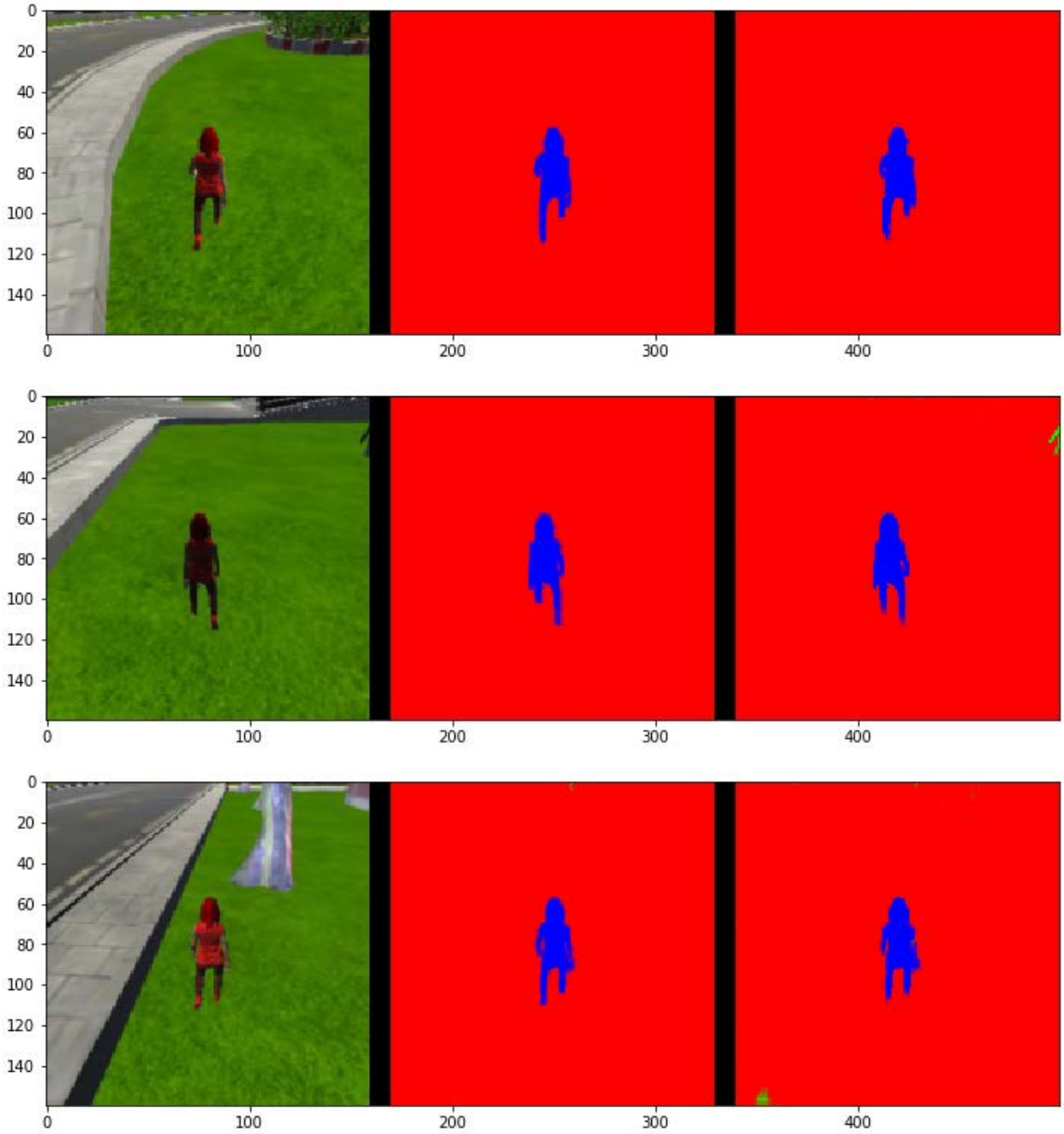
```
# This score measures how well the neural network can detect the target from far away
true_pos3, false_pos3, false_neg3, iou3 = scoring_utils.score_run_iou(val_with_targ, pred_with_targ)
```

```
number of validation samples intersection over the union evaulated on 322
average intersection over union for background is 0.996499004335
average intersection over union for other people is 0.453317615995
average intersection over union for the hero is 0.215140560149
number true positives: 124, number false positives: 2, number false negatives: 177
```

Final Score : 0.41

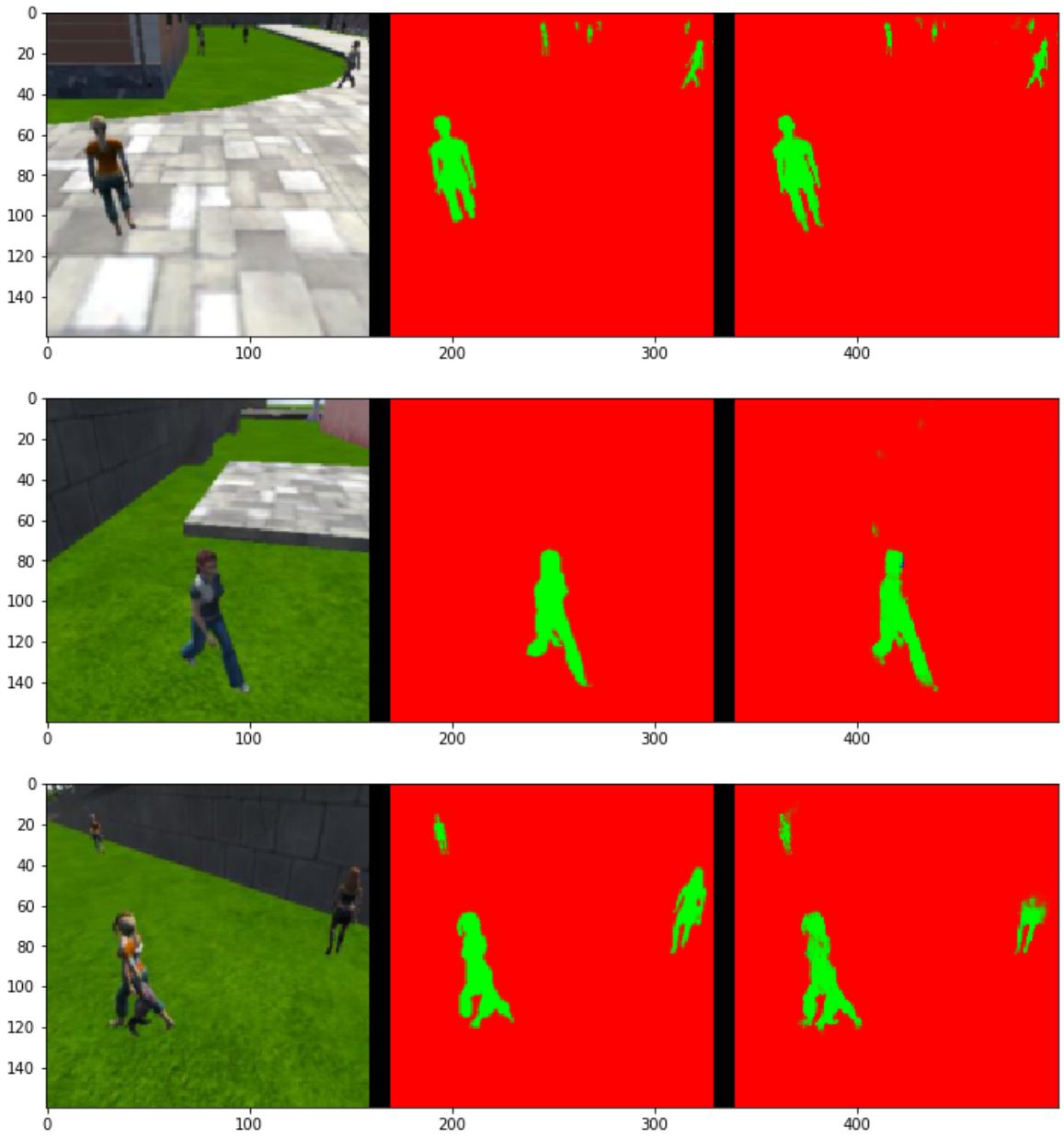
Images while following the target

This works pretty good while there's a tiny errors exists on the third image.



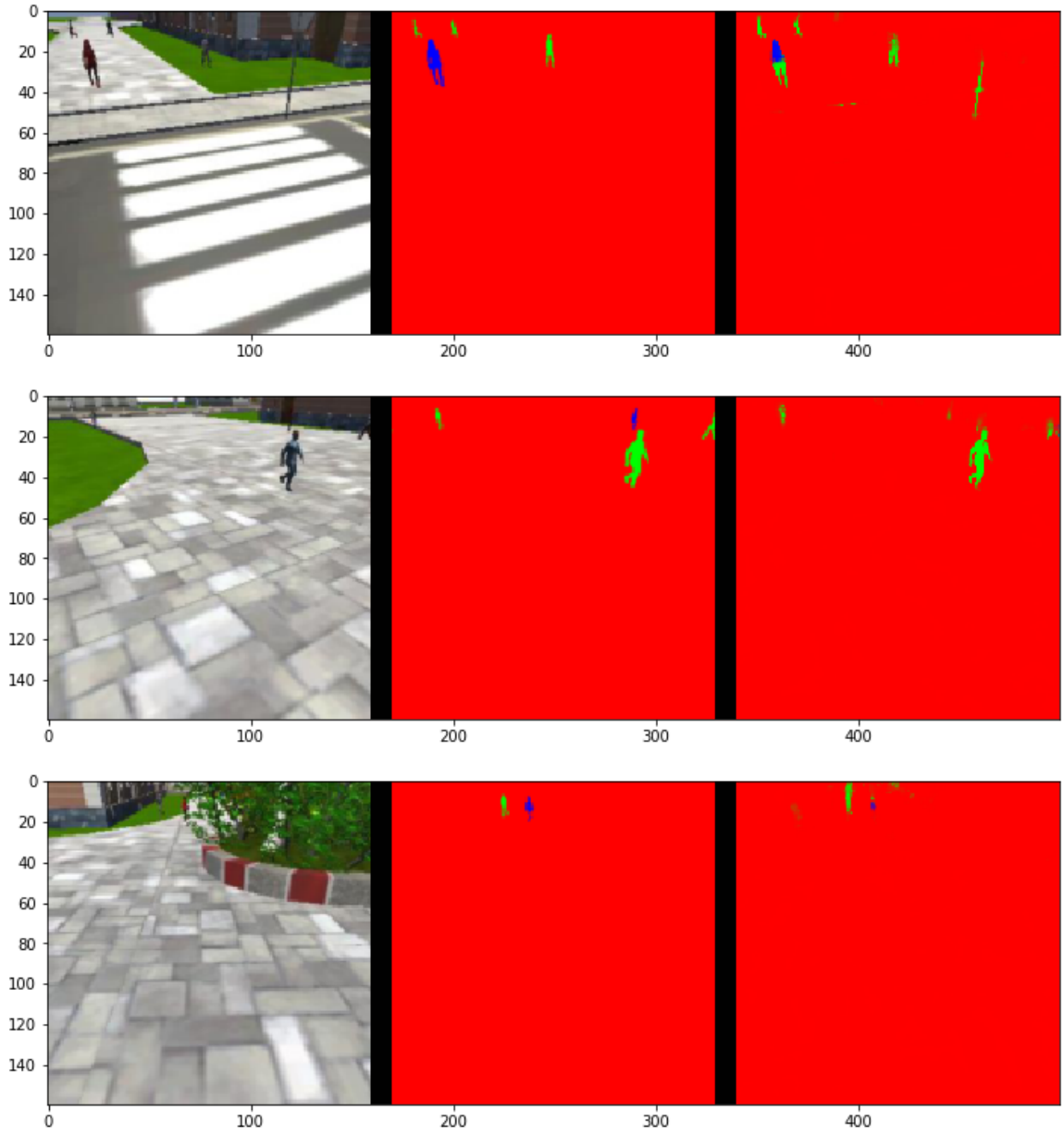
Images while at Patrol without target

As you can see from the image below, this works pretty well!



Images while at patrol with target

This is challenging when detecting the target at very far away from the quadrotor vision. I feel surprise it works to detect the target.



Future Enhancement

Nothing is perfect, there's always lots of fun works to improve.

1. Improving the quality of data
2. Increase the numbers of data
3. Better Hyperparameters
4. Data augmentations
5. Model Architecture enhancement
6. Utilize more Visualize tools for further analysis
7. Following others target likes dogs, cats or other animals.